A Survey of Techniques for Information Flow Analysis and Modeling of Faceted Values for Controlling Malicious JavaScript

Konstantinos Xirogiannopoulos and Benjamin Bengfort

Department of Computer Science
University of Maryland
{kostasx,bengfort}@cs.umd.edu

November 6, 2015

Abstract

(1) a revised summary of your research based on the proposal; (

1. Notes from [1]
2. is important to prevent the results of the computation from leaking even partial information about confidential inputs
3. The ability to track implicit flows accurately is one of the major advantages of static enforcement of information-flow control
4. It is a common assumption in language-based work on information flow that information-flow policies are known statically, at compile time. This is not a realistic assumption for a large computing system.
5. The difficulty in adding dynamic labels is that because they are computed at run time, they create an additional information channel that must be controlled through the type system
6. It is important that security static analyses do not reject too many secure programs, even though they are necessarily conservative
7. Security-type systems: Type systems are attractive for implementing static security analyses. It is natural to augment type annotations with security labels. Type systems allow for compositional reasoning, which is a necessity for scalability when applied to larger programs. Many well developed features of type systems have been usefully applied to security analysis
8. Implicit flows signal information through the control structure of a program. • *Termination channels* signal information through the termination or nontermination of a computation. • *Timing channels signal* information through the time at which an action occurs rather than through the data associated with the action. The action may be program termination; that is, sensitive information might be obtained from the total execution time of a program. • *Probabilistic channels* signal information by changing the probability distribution of observable data. These channels are dangerous when the attacker can repeatedly run a computation and observe its stochastic properties. • *Resource exhaustion channels* signal information by the

possible exhaustion of a finite, shared resource, such as memory or disk space. • *Power channels* embed information in the power consumed by the computer, assuming that the attacker can measure this consumption.

9. no-sensitive-upgrade semantics [39, 5] and the permissive-upgrade semantics guarantee *termination insensitive non-interference.*

Implicit Flows in Information Flow

In today's information age, the software systems we use every day hold all kinds of information about us; some of which highly sensitive. From social media passwords to bank information and credit card numbers; these pieces of highly sensitive data are being passed around and propagated throughout the systems we use, their travel path getting increasingly longer as these systems increase in complexity.

Much work has been done on communicating confidential data, such as cryptography, access control and firewalls. For example, sending a cryptographic message from one machine to another is usually considered secure; once however the message arrives to its destination and is decrypted, there arises the issue of what happens to this nugget of information as it *flows* through the system to serve its purpose. No guarantees can thus be given about the data after its decryption, regardless of its level of sensitivity.

Data is usually *propagated* through different variables within the system. Values often take complex routes within programs, changing multiple hands (variables) along their usage-cycle. The problem is that there are often variables of different levels of security; variables that interact with potentially public pieces of code, and variables that are limited to specific pieces of code, executed in a secure fashion. *Information flow* studies the ways these values travel within the system, from variable `x` to `y`.

There are two main categories of Information flow; *explicit* and *implicit.* Explicit flows are easy to detect, as they are flows where the value of variable x is somehow explicitly passed into variable y, either by direct or indirect assignment of the sensitive data to a non-secure variable. Implicit flows on the other hand are much more difficult to pinpoint as the sensitive data is not directly ever passed into a non-secure variable, but rather the information can be inferred by some sort of *side channel.* Examples of this include being able to infer the value of a confidential variable by the amount of time it takes for the program to run or the value a low confidentiality variable takes in a branch involving the value of the highly confidential variable.

The most desirable policy in this case is that of *Non-Interference*; which states that there should absolutely be no difference between two executions of the same program if the only thing that changes between these executions are the program's secret values. This policy however is too strict and not practically possible to be enforced in real-world applications. Another way to phrase noninterference is *no data visible to other users is affected by confidential data.* Confidential data may not interfere with publicly observable data. Noninterference can be naturally expressed by semantic models of program execution.

There exists however a more relaxed notion of non-interference called *termination insensitive non-interference.* This guarantee states that private inputs do not influence public outputs, *however* private variables can influence program termination. This channel of obtaining information about high confidentiality variables nevertheless is only open to brute force attacks.

Static Information Flow Control

0.1 Security Type System

Semantics based security for guaranteeing noninterference. [1]

2) a description of the research problem you set out to tackle;

Faceted Values towards Javascript Security

(3) a description of your solution including any artifacts produced from your work;

Evaluation of the Faceted Values Approach

(4) an evaluation of your research providing evidence for how it succeeds or fails in solving your research problem;

Future Work

(5) a summary of future challenges. Include a bibliography of any relevant related work.

References

[1] Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *Selected Areas in Communications, IEEE Journal on*, 21(1):5–19, 2003.